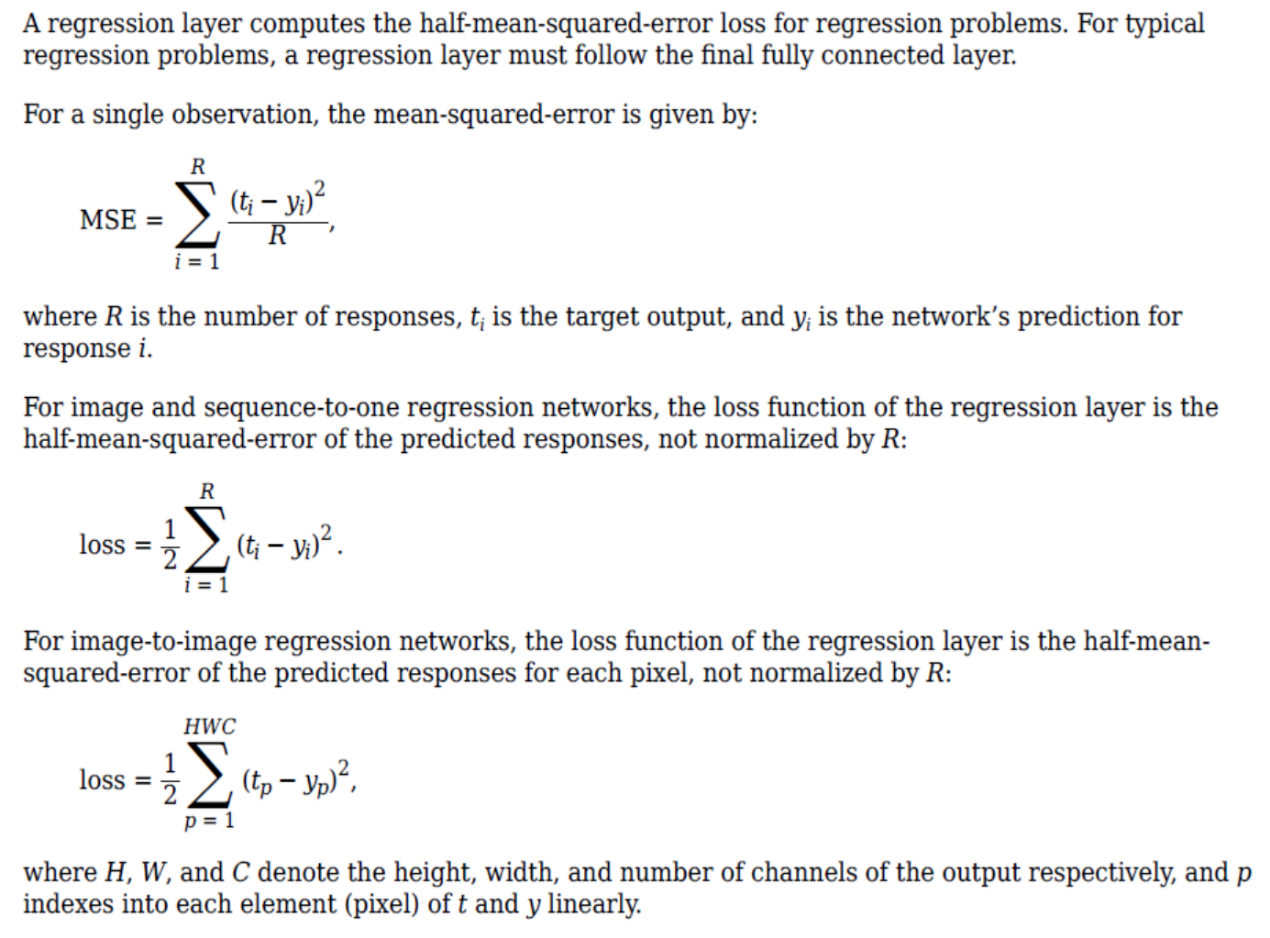
**Matlab神经网络**

**1.RMSE：**MSE的sqrt，将观测值与期望值的MSE算出来取平均，再开平方得到RMSE。

例如，若batchsize=20，即每次从cell中取20个元胞作为一个iteration，每个元胞里是一个(M,N)的矩阵，20个元胞对应的输出为20个y\_hat（即下图中的ti）。那么MSE就是将这20个ti与期望值yi逐元素相减、取平方、对元素求平均。得到的MSE再开根号即为RMSE。得到的MSE、RMSE是针对每个元素的MSE、RMSE。



图形用户界面, 文本

描述已自动生成

**2.全连接层**，y=w\*x+b，若是用来做信道估计，那么实际上w就是估计出来的信道矩阵H，因此可将x变成toeplitz阵，则w=[w1,w2,…wn]，等于H=[H1,H2,…,Hn]。

**文件夹注释**

data\_save文件夹:

2.21文件夹：储存的数据是没有加导频、没有同步的数据，储存的是25M发射数据和150M接收数据。经过了上采样到160M、过信道、下采样到150M的过程，信道为随机数。

2.22文件夹：储存的数据是加了导频、同步后的数据，但采集的数据里面是把导频去掉了，储存的是25M发射数据和150M接收数据。经过了上采样到160M、过信道、下采样到150M的过程，信道为fir滤波器。

2.23\data文件夹：与2.22文件夹相同，只是用的滤波器参数不一样

2.23\data2文件夹：储存的数据是加了导频、同步后的数据，但采集的数据里面是把导频去掉了，储存的是25M发射数据和150M接收数据。经过了上采样到160M、过信道、下采样到150M的过程，但信道没有做任何事（即信道相当于一个冲激函数），即160M发送数据=160M接收数据，signal\_pass\_channel=signal\_send。

2.26文件夹：随机生成一段随机数当作发送数据，再生成一段长度为9的随机数当作信道，发送数据与信道卷积得到接收数据，最后将发送数据和接收数据存储下来。即上面几个文件夹里存储的数据是25M的发送数据、150M的接收数据，而2.26文件夹存储的发送数据和接收数据的速率相同。

light\_data\result文件夹（已删除）：存放的结果是对应 2.25调试日志.docx 中2.1.4、2.1.4.2 的调试结果。

light\_data\_2.28文件夹：

\data：储存了光路信道25M发送数据、150M接收数据、精采样点，发送数据是8pam数据。

\result：见当天的调试日志。

light\_data\_3.4文件夹：

\data：储存了光路信道25M发送数据、150M接收数据、精采样点。与2.28不同的是，发送数据是均匀分布的随机数据。

\result：见当天的调试日志。

light\_data\_3.8文件夹：

\data：储存了光路信道10M发送数据、60M接收数据、精采样点。与3.4不同的是，发送数据的采样率是10M，但仍然是均匀分布的随机数据。

\data\rand\_bias0.3：信号的偏置电流是0.3A。

\data\rand\_bias0.6：信号的偏置电流是0.6A。

\result：见当天的调试日志。

light\_data\_3.9文件夹：

\data：储存了光路信道10M发送数据、60M接收数据、精采样点，数据是均匀分布的随机数据。与3.8不同的是，3.8是按照snr来划分，3.9是按照发送信号幅度来划分。

\data\rand\_bias0.3：信号的偏置电流是0.3A。

\data\rand\_bias0.6：信号的偏置电流是0.6A。

\result：见当天的调试日志。

light\_data\_3.10文件夹：

\data：储存了光路信道10M发送数据、60M接收数据、精采样点、发送信号幅度归一化时的归一化因子，数据是均匀分布的随机数据。与之前不同的是，3.10还存储了光路发送信号用来幅度归一化的归一化因子。

\data\rand\_bias0.3：信号的偏置电流是0.3A。

\result：见当天的调试日志。

light\_data\_3.11文件夹（已删除）：

\data：储存了光路信道10M发送数据、150M接收数据、精采样点、发送信号幅度归一化时的归一化因子，数据是均匀分布的随机数据。

\data\rand\_bias0.3：信号的偏置电流是0.3A。

\result：见当天的调试日志。

light\_data\_3.16文件夹：

\data：储存了光路信道10M发送数据、60M接收数据、精采样点、发送信号幅度归一化时的归一化因子，数据是均匀分布的随机数据。与之前不同的是，这次采集的数据前面幅度信号都采集的很少，大部分数据采集的是幅度最大的信号的数据。以此来实验用大幅度数据训练出来的网络能否适用于小幅度数据。

\data\rand\_bias0.3：信号的偏置电流是0.3A。

\result：见当天的调试日志。

light\_data\_3.17文件夹：

\data：储存了光路信道10M发送数据、60M接收数据、精采样点、发送信号幅度归一化时的归一化因子，初始分布为bpsk。此数据用于mix amp训练方式，为的是弥补之前mix amp训练时高幅度数据数量不足的情况。

\data\rand\_bias0.3：信号的偏置电流是0.3A。

\result：见当天的调试日志。

**程序注释**

data.m：用来生成简单的测试数据并存到文件里，用于简单的网络的训练。

dnn\_all\_amp.m（之前的test\_dnn3.m）：与 test\_dnn2.m 功能类似，只是 test\_dnn2.m 的训练数据只包括了三个不同的snr，而 test\_dnn3.m 包含了很多个不同的snr。

dnn\_all\_bias.m：用于对不同速率的收发信号进行深度学习训练，该程序的训练数据是不同bias、不同amp信号的拼接。对所有bias、amp训练一个网络。

dnn\_all\_data.m：先用一部分数据训练网络，训练好之后将网络保存，然后加载新的数据接着训练保存好的网络，这样可以做到增加训练样本数据量。4.10将其改为自定义训练的程序。

dnn\_all\_data\_custom.m：先用一部分数据训练网络，训练好之后将网络保存，然后加载新的数据接着训练保存好的网络，这样可以做到增加训练样本数据量。4.10将其改为自定义训练的程序。与dnn\_all\_data\_custom\_2.m的区别是，该程序单次训练放进网络里的amp只有两种，bias只有所有bias的一半。而dnn\_all\_data\_custom\_2.m是将所有amp-bias组合全部放进网络中。

dnn\_all\_data\_custom\_2.m：先用一部分数据训练网络，训练好之后将网络保存，然后加载新的数据接着训练保存好的网络，这样可以做到增加训练样本数据量。该程序单次训练使用的训练数据是所有amp-bias组合，由于内存限制，每个amp-bias组合中包含的cell数量都比dnn\_all\_data\_custom.m少。

dnn\_all\_data\_default.m：该程序单次训练放入的训练数据包含的amp只有两种，bias只有所有bias的一半，即和dnn\_all\_data\_custom.m一样。但是dnn\_all\_data\_custom.m的训练方式是自定义训练，而dnn\_all\_data\_default.m是用Matlab自带的训练函数训练。

dnn\_all\_data\_type\_3.m：用Matlab自带的训练函数训练，但单次训练使用的训练数据是所有amp-bias组合。

dnn\_high\_amp.m（之前的test\_dnn4.m）：与test\_dnn3.m不同的是，此程序用的训练数据是幅度较大的几个数据，而不是全部幅度的数据，以此来试验用大幅度数据训练出来的网络能否适用于小幅度数据。

dnn\_option.m：该程序中存储了网络结构和训练选项，用来给dnn\_all\_data\_default.m使用。

dnn\_single\_amp.m（之前的test\_dnn.m）：用于对不同速率的收发信号进行深度学习训练，与test\_dnn2.m、test\_dnn3.m不同的是，该程序的训练数据是单一snr的信号，而不是不同snr信号的拼接。

dnn\_single\_bias.m：用于对不同速率的收发信号进行深度学习训练，该程序的训练数据是相同bias、不同amp的信号，而不是不同bias信号的拼接。对每个bias都训练一个网络出来。

dnn\_train\_custom.m：自定义训练，和dnn\_train\_custom\_2.m的区别是该程序没有中止训练的程序。

dnn\_train\_custom\_2.m：自定义训练，和dnn\_train\_custom.m的区别是该程序有中止训练的程序。

dnn\_train\_default.m：用系统自带的训练函数进行训练，给dnn\_all\_data\_default.m使用。

dnn\_trainedNet.m：对已经训练好的网络测试性能。

load\_bias\_amp.m：用来把**全部**amp的**全部**bias的数据读取出来。

load\_bias\_amp\_custom.m：给dnn\_all\_data\_custom.m使用，每次读取两个amp、一半bias的数据出来。

load\_bias\_amp\_custom\_2.m：给dnn\_all\_data\_custom\_2.m使用，每次读取所有amp-bias组合的数据出来。

load\_data.m：用来从文件中读取**指定数量**的数据。

load\_data\_all.m：用来从文件中读取**全部数量**的数据。

ls\_amp.m：(之前的ls\_esi2.m)用来对信号进行ls估计，如果有1000组信号，那么采用其中**800**组用来做ls估计，再采用剩下**200**组来评估ls性能。

ls\_bias.m：与ls\_amp.m功能类似，只是该程序不仅针对不同的amp，还针对不同的偏置电流bias。

ls\_esi.m：用来对信号进行ls估计，如果有1000组信号，那么采用其中一组用来做ls估计，再采用另外一组来评估ls性能（mse）。

optimal\_nmse.m：用于计算不同snr时的最佳NMSE。

split\_bias.m：用来将单一amp的所有bias随机打散、拼接。

split\_data\_custom.m：给dnn\_all\_data\_custom.m使用，根据输入的amp、bias范围，输出合适的amp-bias组合，告诉dnn\_all\_data\_custom.m该次训练要使用哪些amp-bias组合的数据。

split\_data\_custom\_2.m：给dnn\_all\_data\_custom\_2.m使用，根据输入的amp、bias范围，将所有的amp-bias随机组合。

test\_dnn.m：用于对同速率的收发信号进行深度学习训练。

test\_dnn2.m：用于对不同速率的收发信号进行深度学习训练，比如发送数据25M，接收数据150M。并且训练的数据是不同snr（即不同幅度的发送信号）的发送信号拼接在一起，训练出来的网络是适用于不同snr发送信号的网络。